Computing 21, 295—322 (1979)

Computing
© by Springer-Verlag 1979

Clique Detection for Nondirected Graphs: Two New Algorithms

L. Gerhards and **W. Lindenberg**, St. Augustin

Received March 20, 1978; revised September 14, 1978

Abstract — Zusammenfassung

Clique Detection for Nondirected Graphs: Two New Algorithms. Making use of special tree search algorithms the present paper describes two new methods for determining all maximal complete subgraphs (cliques) of a finite nondirected graph. In both methods the blockwise generation of all cliques induces characteristic properties, which guarantee an efficient calculation of special clique subsets, especially the set of all cliques of maximal length. Moreover, by their structure both algorithms allow to calculate the complete clique set by parallel processing. The algorithms have been tested for many series of characteristic graphs and compared with the algorithm of Bron-Kerbosch (Algorithm 457 of CACM) the most efficient algorithm which is known to the authors.

Cliquenbestimmung in ungerichteten Graphen: Zwei neue Algorithmen. Die folgende Arbeit enthält zwei neue Algorithmen zur Bestimmung der Menge sämtlicher maximaler vollständiger Untergraphen (Cliquen) eines endlichen ungerichteten Graphen. Die Methoden verwenden spezielle Baumsuchalgorithmen. Die blockweise Erzeugung aller Cliquen führt zu charakteristischen Eigenschaften der Algorithmen, die eine effiziente Berechnung spezieller Untermengen von Cliquen, u. a. die Menge aller Cliquen von maximaler Länge, ermöglichen. Überdies erlaubt die Struktur beider Algorithmen die Berechnung der vollständigen Cliquenmenge auf parallel arbeitenden Rechnern. Die Algorithmen wurden an umfangreichen Serien charakteristischer Graphen getestet und mit dem wirksamsten der den Autoren bekannten Algorithmen, dem Algorithmus von Bron-Kerbosch (Algorithm 457 of CACM), verglichen.

Introduction

A maximal complete subgraph of a (nondirected) graph — that is a complete subgraph which is not contained in any other complete subgraph — is called a clique.

It is well known that the determination of the cliques of a graph (respectively the determination of the internally stable sets) is an important problem for it occurs in many diverse applications as in cluster analysis, classification theory, graph coloring, information retrieval systems, disposal systems, biological systems and many socionomic concepts.

Therefore, in the last ten years, many clique detection algorithms have been developed [1], [2], [4], [5], [6], [8], [11], [12] which — roughly spoken — can be divided into two classes. The first class contains those algorithms, which build up a clique of the graph step by step, constructing set systems of vertices of the graph or systems of matrices which are to be stored in long lists and which are changed during the calculation procedure. The second class includes those algorithms which do not need the information of any clique after its construction and notation. Therefore, this class consists of space-saving and time-efficient clique detection methods (cf. [2], [6], [12]) among which the algorithm [2] of Bron-Kerbosch is on the average close to the best possible by its structure.

The new algorithms described in the present paper belong to the second class. Comparing their computing time with that of the algorithm [2], the new algorithms are nearly of the same efficiency. Moreover, in the case of graphs of high symmetric structure or of sparse adjacency matrices the developed algorithms are most efficient. The basic concept of the new algorithms is to construct special subsystems of cliques each of which is related to a fixed vertex of the graph and whose totality builds up the whole set of cliques of the graph. These clique systems are found by different tree search techniques which guarantee in a simple way, that a clique will be constructed only once and which allow cutting off branches that cannot lead to a clique. The theoretical background of the techniques used is explained in the sections A—D of the paper.

Additionally, the independent blockwise generation of the total clique set of the graph involves some remarkable characteristic properties (described in section E) of the new algorithms, as:

— the possibility of the use of parallel processors during the calculation procedure;
— the calculation of special clique sets whose elements contain a prescribed set of vertices;
— the computation of the cliques of maximal length.

Numerous test results obtained with the new algorithms were compared with the results of algorithm [2]. A collection of these results for some characteristic series of graphs is finally given in section F by time-tables and diagrams.

A. Graph-Theoretical Definitions and Results

1. Notations and Definitions

1.1 A *nondirected graph* $G=(V, E)$ consists of a set of vertices V and a set of edges E, where $E \subseteq S(V)$ is a subset of the set $S(V)$ of *unordered* pairs $(v_i, v_j) := (v_j, v_i)$ of *different* vertices $v_i, v_j \in V$. $S(V)$ can be obtained from $(V \times V) \backslash \Delta$ where $\Delta = \{(v_i, v_i) \in V \times V / v_i \in V\}$ by an identification of the elements (v_i, v_j) and (v_j, v_i) $(i \neq j)$.

1.2 A *subgraph* of $G=(V, E)$ is a graph $G'=(V', E_{V'})$ where $V' \subseteq V$ and $E_{V'} = E \cap S(V')$.

The following Lemma is obvious:

Lemma 1.2.1: *If $G_1=(V_1, E_{V_1})$ and $G_2=(V_2, E_{V_2})$ are subgraphs of the graph $G=(V, E)$, then G_2 is a subgraph of G_1 iff $V_2 \subseteq V_1$.*

1.3 Two vertices v_i, v_j of $G=(V, E)$ are said to be *connected* in G (denoted by $\underset{G}{\sim}$) if $(v_i, v_j):=(v_j, v_i)\in E$. If $G'=(V', E_{V'})$ is a subgraph of $G=(V, E)$, we immediately obtain the implication $v_i \underset{G'}{\sim} v_j \Rightarrow v_i \underset{G}{\sim} v_j$.

1.4 A *complete graph* $G=(V, E)$ is one in which each vertex is connected to every other vertex: $v_i \underset{G}{\sim} v_j$ for all $v_i, v_j \in V (i\neq j)$.

1.5 If $V=\{v_1, \dots, v_n\}$ is the vertex set of $G=(V, E)$, every vertex $v_i \in V$ can be represented by its subscript i and we obtain a total ordering on V:

$$v_i<v_j \Leftrightarrow i<j \ (i\neq j).$$

1.6 A graph $G=(V, E)$, $|V|=n$ can be represented by a symmetric $(n\times n)$-matrix (called the adjacency matrix of G):

$$M(G):=(m_{ij})_{n\times n}, \text{ where } \begin{cases} m_{ij}=1, \text{ if } i \underset{G}{\sim} j \\ m_{ij}=0, \text{ if } i \underset{G}{\not\sim} j \end{cases}.$$

If $G'=(V', E_{V'})$ is a subgraph of $G=(V, E)$, the matrix $M(G')$ can be obtained from $M(G)$ by deleting those rows and columns of $M(G)$ which correspond to the vertices of $V\backslash V'$.

1.7 A *maximal complete subgraph* $C=(V', E_{V'})$ of the graph $G=(V, E)$ is called a *clique* of G. The set of all cliques of G may be denoted by L_G.

1.8 For an arbitrary nondirected graph $G=(V, E)$, the graph $G'=(V', E')$ defined by $V'=V$, $E'=E\cup\Delta$ has the same set of cliques as G provided that a clique is regarded only as its set of vertices. For computational reasons, in the following we may restrict our attention strictly to graphs $G=(V, E)$ where $E=E'\cup\Delta$ and $E'\subseteq S(V)$ except in section D where G is a nondirected graph in the sense of the definition 1.1.

2. *Neighborhoods in a Graph*

2.1 Let $G=(V, E)$ be a graph. Then the two subsets of V

$$N_G(i):=\{j\in V/i \underset{G}{\sim} j\} \quad \text{and} \quad N_G^{\leq}(i):=\{j\in N_G(i)/j\leq i\} \tag{2.1}$$

are called the i-neighborhood (reduced i-neighborhood) of the vertex $i\in V$ *in* G respectively.

Lemma 2.1.1: *Suppose that $G'=(V', E_{V'})$ is a subgraph of $G=(V, E)$ and that $i\in V'$, then:*

$$N_{G'}(i)=N_G(i)\cap V' \quad \textit{and} \quad N_{G'}^{\leq}(i)=N_G^{\leq}(i)\cap V'. \tag{2.2}$$

Proof: Suppose $j\in N_{G'}(i)$. Then it follows that $j\in V'$ and $i \underset{G'}{\sim} j$ and according to 1.3 we have $i \underset{G}{\sim} j$. Since $V'\subseteq V$, we obtain that $j\in N_G(i)$ and therefore $j\in V'\cap N_G(i)$, hence $N_{G'}(i)\subseteq N_G(i)\cap V'$. Suppose now that $j\in N_G(i)\cap V'$. Then $j\in V'$ and $i \underset{G}{\sim} j$. Since $i\in V'$ we obtain $i \underset{G'}{\sim} j$, and consequently $j\in N_{G'}(i)$. Therefore, the inclusion $N_G(i)\cap V'\subseteq N_{G'}(i)$ is valid. Similar for the second relation.

2.2 For the graph $G=(V, E)$ of $|V|=n$ vertices, one defines for $1\leq i\leq n$ the following subgraphs of G:

$$S_i := (N_G(i), E_{N_G(i)}) \text{ and } S_i^{\leq} := (N_G^{\leq}(i), E_{N_G^{\leq}(i)}) \tag{2.3}$$

which play an important role in the construction of L_G.

Since $N_G^{\leq}(i) \subseteq N_G(i)$, by Lemma 1.2.1, $S_i^{\leq}$ is a subgraph of S_i.

B. Theoretical Background of the Algorithm A 1[1]

3. Characterization of Cliques by Neighborhoods

3.1 Let $G' = (V', E_{V'})$ be a subgraph of $G = (V, E)$ and $Q \subseteq V'$. Then the following Theorem gives a necessary and sufficient neighborhood condition for the subgraph $C = (Q, E_Q)$ of G to be a clique of G':

Theorem 3.1.1: $C \in L_{G'} \Leftrightarrow Q = \bigcap_{j \in Q} N_{G'}(j)$.

Proof: $\Rightarrow$: Assume $C \in L_{G'}$. Then, since C is a complete graph $i \in Q$ implies $i \in \bigcap_{j \in Q} N_{G'}(j)$, hence $Q \subseteq \bigcap_{j \in Q} N_{G'}(j)$. If $Q \subset \bigcap_{j \in Q} N_{G'}(j)$, there exists an element $k \in \bigcap_{j \in Q} N_{G'}(j)$ such that $k \notin Q$. $Q \cup \{k\}$, however, generates a complete subgraph of G, but this contradicts the assumption that C is maximal. Therefore, we obtain $Q = \bigcap_{j \in Q} N_{G'}(j)$.

$\Leftarrow$: Assume $Q = \bigcap_{j \in Q} N_{G'}(j)$. Suppose further that C is not a complete subgraph of G'. Then there are $i, k \in Q$ such that $i \underset{G'}{\not\sim} k$. Thus i or k is not contained in $\bigcap_{j \in Q} N_{G'}(j)$. But this contradicts the assumption and therefore C is complete. Now let C be complete but not maximal in G'. Then $\bigcap_{j \in Q} N_{G'}(j)$ contains an element $i \notin Q$ also contradicting the assumption.

Theorem 3.1.2: *Let $C = (Q, E_Q)$ be a subgraph of $G = (V, E)$, then*:

$$C \in L_{S_i} \Leftrightarrow C \in L_G \text{ and } i \in Q.$$

Proof: Using Lemma 2.1.1, we obtain:

$$\bigcap_{j \in Q} N_{S_i}(j) = \bigcap_{j \in Q} N_G(j) \text{ if } i \in Q. \tag{3.1}$$

$\Rightarrow$: Suppose $C \in L_{S_i}$. Then $Q = \bigcap_{j \in Q} N_{S_i}(j)$ by Theorem 3.1.1. Since C is a subgraph of S_i, it follows that $Q \subseteq N_G(i)$ by Lemma 1.2.1. Because $i \underset{S_i}{\sim} j$ for all $j \in N_G(i)$, one concludes $i \in Q$ and by (3.1) we have $Q = \bigcap_{j \in Q} N_G(j)$, hence $C \in L_G$ by Theorem 3.1.1.

$\Leftarrow$: Suppose $C \in L_G$ and $i \in Q$. Then $Q = \bigcap_{j \in Q} N_G(j)$ by Theorem 3.1.1. Since $i \in Q$, by (3.1) we obtain $Q = \bigcap_{j \in Q} N_{S_i}(j)$ and therefore by Theorem 3.1.1, $C \in L_{S_i}$.

[1] This algorithm has been developed by the first author.

4. *i-Systems of Cliques and Their Characterization*

4.1 For each vertex $i \in V$ of a finite graph $G=(V, E)$ and ordered vertex set V let

$$B_i := \{C=(Q, E_Q) \in L_G / i \in Q \subseteq N_G^{\leq}(i)\} \tag{4.1}$$

be the subset of all cliques of G which contain only vertices $j \leq i$ of V.

Theorem 4.1.1: *Let $C=(Q, E_Q)$ be a clique of $G=(V, E)$. Then:*

$$C \in B_i \Leftrightarrow Q \subseteq N_G^{\leq}(i) = N_{S_i}^{\leq}(i) \quad and \quad Q = \bigcap_{j \in Q} N_{S_i}(j).$$

Proof: $\Rightarrow$: If $C \in B_i$ the relation $Q \subseteq N_G^{\leq}(i)$ immediately follows from the definition of B_i. Since i is connected with every element of $N_G^{\leq}(i)$, we have $i \in Q$ and by Theorem 3.1.2 $C \in L_{S_i}$, hence $Q = \bigcap_{j \in Q} N_{S_i}(j)$ by Theorem 3.1.1.

$\Leftarrow$: Conversely, if $Q = \bigcap_{j \in Q} N_{S_i}(j)$, by Theorem 3.1.1 it follows that $C \in L_{S_i}$ and by Theorem 3.1.2 we obtain $i \in Q$ and $C \in L_G$. Together with the condition $Q \subseteq N_G^{\leq}(i)$, we conclude that $C \in B_i$.

Theorem 4.1.2: *The following statements are equivalent*:

(a) $B_i = \emptyset$.

(b) *For all cliques $C=(Q, E_Q) \in L_{S_i^{\leq}}$ there exists an element $k \in N_G(i) \backslash N_G^{\leq}(i)$ such that $k \underset{S_i}{\sim} j$ for all $j \in Q$.*

(c) $Q \neq \bigcap_{j \in Q} N_{S_i}(j)$ *for all subsets* $Q \subseteq N_G^{\leq}(i)$.

Proof: (a) $\Rightarrow$ (b): If $B_i = \emptyset$ every clique of $S_i^{\leq}$ can be imbedded in a complete subgraph of S_i which contains a vertex $k \in N_G(i) \backslash N_G^{\leq}(i)$.

(b) $\Rightarrow$ (c): By Theorem 3.1.1, $Q \subseteq N_G^{\leq}(i)$ is the generating set of a clique of S_i iff $Q = \bigcap_{j \in Q} N_{S_i}(j)$. Since condition (b) means that every clique of $S_i^{\leq}$ can be imbedded in a complete subgraph of S_i, condition (b) implies (c).

(c) $\Rightarrow$ (a): If (c) is satisfied then by Theorem 4.1.1 $B_i = \emptyset$.

Corollary 4.1.3: $B_i = \emptyset$, *if* $N_G(i) \backslash N_G^{\leq}(i)$ *contains an element* k *such that* $k \underset{S_i}{\sim} j$ *for all* $j \in N_G^{\leq}(i)$.

Proof: The condition implies condition (b) of Theorem 4.1.2.

4.2 If $B_i \neq \emptyset$, there exists a well defined greatest complete subgraph $H_i^{\leq}$ of $S_i^{\leq}$ whose vertex set $T_G^{\leq}(i) \subseteq N_G^{\leq}(i)$ consists of all elements of $N_G^{\leq}(i)$ which are connected with each vertex of $N_G^{\leq}(i)$ itself. $H_i^{\leq}$ always exists, since at least $i \in T_G^{\leq}(i)$. Moreover, $H_i^{\leq}$ is a subgraph of each clique of B_i.

By these remarks and Theorem 4.1.2 we obtain the following Reduction Theorem:

Theorem 4.2.1: *Every subset $Q = T_G^{\leq}(i) \cup K \subseteq N_G^{\leq}(i)$ is the generating vertex set of a clique of B_i iff $K \subseteq N_G^{\leq}(i) \backslash T_G^{\leq}(i)$ is the vertex set of a clique of the subgraph S_i^* of S_i generated by $(N_G^{\leq}(i) \backslash T_G^{\leq}(i)) \cup R_i$ where R_i is the subset of $N_G(i) \backslash N_G^{\leq}(i)$ whose elements are connected (say in G) with all elements of $T_G^{\leq}(i)$.*

Theorem 4.2.1 is the basic theorem for the development of algorithm $A1$ described in the following section. The reduction to the relatively small subgraph S_i^* effects the efficiency of this method. In version 1 of algorithm $A1$ Theorem 3.1.1 is the fundamental theorem for the decision, if a vertex set of S_i^* is a clique of this subgraph. Version 2 of A 1 needs another clique criterion, which will be described in section C., 6.3.

C. A Short Description of the Implementation of Algorithm A 1

5. Basic Structure of A 1

5.1 Some Notation

5.1.1 Every graph $G=(V, E)$ regarded in $A1$ is represented in the computer by its adjacency matrix $M(G)$, which is symmetric and reflexive (see 1.6 and 1.8). If $G'=(V', E_{V'})$ is a subgraph of G, then $M(G')$ is obtained from $M(G)$ by deleting all rows and columns of $M(G)$ which correspond to the vertices of $V \setminus V'$.

5.1.2 We introduce the following notation for $M(G)$:

$v_G(i)$	i-th row vector of $M(G)$
$v_G^T(i)$	i-th column vector of $M(G)$
$e_G(i)$	i-th unit vector of length $\mid V \mid$
$m_G(i)$	special mask of the form $(\underbrace{1 \ldots\ldots 1}_{i\text{-times}} \underbrace{0 \ldots\ldots 0}_{(\mid V \mid - i)\text{-times}})$

5.2 The Algorithm CLIQUE

5.2.1 The algorithm CLIQUE consists of an iterative procedure which successively determines the i-systems B_i of cliques of $G=(V, E)$, $(1 \leq i \leq n = \mid V \mid)$.

5.2.2 In testing the sufficient condition of Corollary 4.1.3 by Boolean operations on $M(G)$, in CLIQUE will be examined if there exists an i-system B_i $(1 \leq i \leq n)$:

$$B_i = \emptyset \Leftrightarrow (v_G(i) \wedge m_G(i)) \wedge v_G(j) = v_G(i) \wedge m_G(i) \text{ for some } j \in V, j > i.^2$$

5.2.3 $B_i = \{i\} \Leftrightarrow v_G(i) = e_G(i)$ (i is an isolated vertex).

5.2.4 The vertex set $T_G^{\leq}(i)$ of the graph $H_i^{\leq}$, which is a subgraph of every clique of B_i, can be calculated on $M(S_i) \leftrightarrow S_i(N_G(i), E_{N_G(i)})$. For $j_k \in N_G(i)$, it follows:

$$j_k \in T_G^{\leq}(i) \Leftrightarrow j_k \in N_G^{\leq}(i) \text{ and } m_{S_i}(i) \wedge v_{S_i}(j_k) = m_{S_i}(i).$$

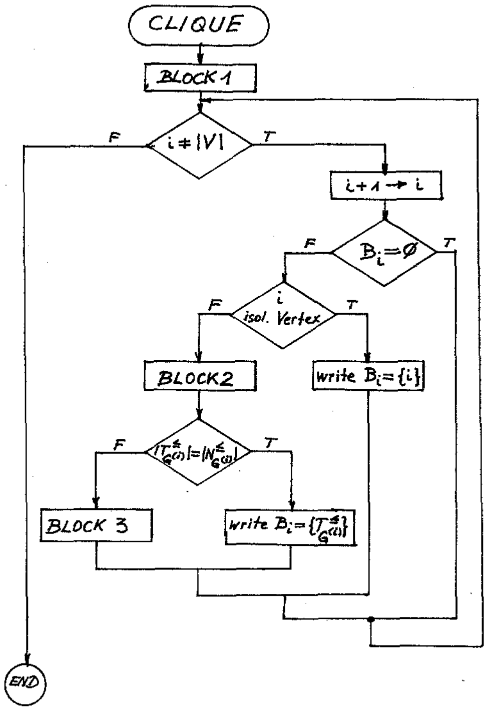
5.2.5 There is only one clique in B_i consisting of the elements of $T_G^{\leq}(i)$ if $\mid T_G^{\leq}(i) \mid = \mid N_G^{\leq}(i) \mid$.

2 The test of this Boolean relation is equivalent to $(v_G(i) \wedge m_G(i)) \vee v_G(j) = v_G(j)$.

5.2.6 If $\tilde{S}_i=(X, E_X)$, $X=(N_G(i)\backslash N_{\tilde{G}}^{\leq}(i)) \cup T_{\tilde{G}}^{\leq}(i)$ and $M(\tilde{S}_i)\leftrightarrow\tilde{S}_i$, then the subset $R_i=\{j_k\in N_G(i)/j_k \underset{\tilde{S}_i}{\sim} l$ for all $l\in T_{\tilde{G}}^{\leq}(i)\}$ defined by Theorem 4.2.1 is computable by testing the following condition:

$$\tilde{S}_i \ni j_k \in R_i \Leftrightarrow j_k \in N_G(i)\backslash N_{\tilde{G}}^{\leq}(i) \text{ and } m_{\tilde{S}_i}(i) \wedge v_{\tilde{S}_i}(j_k)=m_{\tilde{S}_i}(i).$$

5.2.7 Symbolic code of the program CLIQUE.



```
A. procedure CLIQUE:
   begin
     BLOCK 1
     while $i \neq |V|$ do
     begin
       $i \leftarrow i+1$
       if $B_i \neq \emptyset$ then
         begin
           if $i$ isol. Vertex then write $B_i=\{i\}$
           else
             begin
               BLOCK 2
               if $|T_{\tilde{G}}^{\leq}(i)|=|N_{\tilde{G}}^{\leq}(i)|$ then write $B_i=\{T_{\tilde{G}}^{\leq}(i)\}$ else BLOCK 3
             end
         end
     end
   end
```

B. *procedure* BLOCK 1:

```
begin
    read $M(G) \leftrightarrow G=(V,E), (|N_G(i)| \geq |N_G(k)|, (1 \leq i, k \leq n = |V|))$
    $i \leftarrow 0$
end
```

C. *procedure* BLOCK 2:

```
begin
    determine $N_G(i)$ and $M(S_i) \leftrightarrow S_i = (N_G(i), E_{N_G(i)})$ from $M(G)$
    determine $T_G^{\leq}(i)$
    comment For the determination of $T_G^{\leq}(i)$ cf. 5.2.4 in the context
end
```

D. *procedure* BLOCK 3:

```
begin
    determine $R_i$ and $M(S_i^*) \leftrightarrow S_i^* = (X, E_X)$ from $M(S)$
    comment $X$ is defined by $X := (N_G^{\leq}(i) \setminus T_G^{\leq}(i)) \cup R_i$; for determination of $R_i$
      cf. 5.2.6
    determine $\tilde{B}_i = \{K \subseteq N_G^{\leq}(i) \setminus T_G^{\leq}(i) / (K, E_k) \in L_{S_i^*}\}$
    comment The determination of $\tilde{B}_i$ can be done by calling algorithms $K$-CAL 1
    comment or $K$-CAL 2
    determine $B_i = \{Q := K \cup T_G^{\leq}(i) / K \in \tilde{B}_i\}$
end
```

6. *Two Efficient Methods for calculating $\tilde{B}_i$*

6.1 *The Fundamental Graph S_i^**

According to Theorem 4.2.1, for the determination of

$$\tilde{B}_i = \{K \subseteq N_G(i) \setminus T_G^{\leq}(i) / (K, E_k) \in L_{S_i^*}\}$$

we need the adjacency matrix $M(S_i^*)$ of the subgraph $S_i^* = (X, E_X)$ of S_i, where

$$X := (N_G^{\leq}(i) \setminus T_G^{\leq}(i)) \cup R_i \subseteq N_G(i)$$

(for R_i see Theorem 4.2.1). The elements of X will be denoted by $j_1, \ldots, j_{k-1}, j_k, \ldots, j_{|S_i^*|}$, where $j_l < j_{l+1}$ $(l = 1, \ldots, |S_i^*| - 1)$ and

$$\begin{array}{ll} j_l \in N_G^{\leq}(i) \setminus T_G^{\leq}(i) & \text{for } l = 1, \ldots, k-1 \\ j_l \in R_i & \text{for } l = k, \ldots, |S_i^*| \end{array}$$

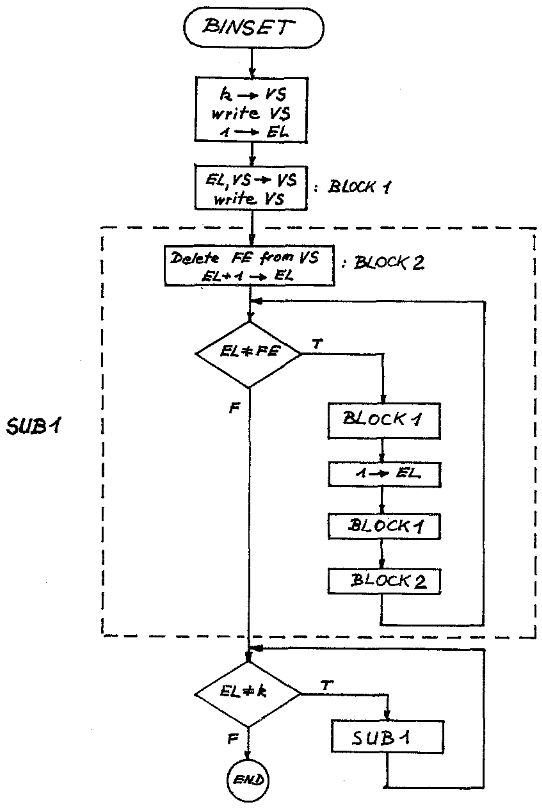
6.2 *The Algorithm K-CAL 1*

6.2.1 The algorithm K-CAL 1 determines the sets $\tilde{B}_i$ $(1 \leq i \leq |V|)$ for the graph $G = (V, E)$ (cf. procedure BLOCK 3 of CLIQUE). It requires a method of selecting suitable subsets of $N_G^{\leq}(i) \setminus T_G^{\leq}(i)$ which are potential generating vertex sets of cliques of S_i^*. If the algorithm is to be efficient, then a subset of $N_G^{\leq}(i) \setminus T_G^{\leq}(i)$ must be generated once and the strategy should be to examine only those sets which are strictly the generating vertex sets of cliques of S_i^*. One method of selecting the subsets $\{j_{i_1}, \ldots, j_{i_r}\}$ of $N_G^{\leq}(i) \backslash T_G^{\leq}(i)$ is to generate all index subsets $\{i_1, \ldots, i_r\}$ of $\{1, \ldots, k-1\}$ ordering them by dual number arithmetic in the following way:

6.2.2 Let I be the set of integers $1, \ldots, k$. Then to any subset $I^* = \{i_1, \ldots, i_s = k\} \subseteq I$ *containing* k there is associated a binary number $D(I^*) = \sum_{l=1}^{s} 2^{i_l - 1}$ and a total ordering $\prec$ of all subsets I^* of I is defined by: $I^* \prec I^{**} \Leftrightarrow D(I^*) < D(I^{**})$. According to this ordering, the subsets I^* of I are selected by the following algorithm:

6.2.3 The algorithm BINSET. Let *FE* be the leading element of a vector *VS* representing the *current* subset.

Algorithm BINSET:



A. *procedure* BINSET:

```
begin
    $VS \leftarrow k$
    write $VS$
    $EL \leftarrow 1$
```

```
        BLOCK 1
        SUB 1
        while $EL \neq k$ do SUB 1
    end

B. procedure SUB 1:
    begin
        BLOCK 2
        while $EL \neq FE$ do
        begin
            BLOCK 1
            $EL \leftarrow 1$
            BLOCK 1
            BLOCK 2
        end
    end

C. procedure BLOCK 1:
    begin
        $VS \leftarrow EL, VS$
        write $VS$
    end

D. procedure BLOCK 2:
    begin
        delete $FE$ from $VS$
        $EL \leftarrow EL+1$
    end
```

6.2.4 During the algorithm K-CAL 1, a reduction of the selected subsets is possible if the current subset

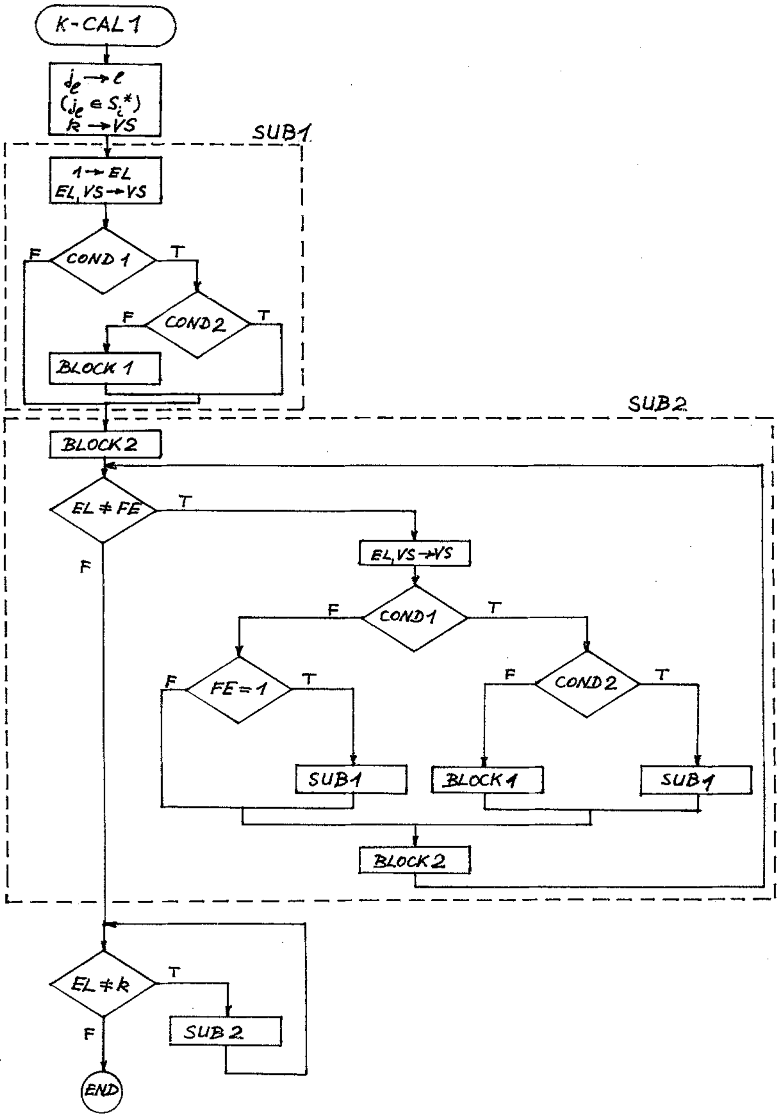
$$\{j_{i_1}, \ldots, j_{i_r}\} \subset \{j_1, \ldots, j_{k-1}\}; \; (j_{i_l} \in N^{\leqq}_{\tilde G}(i) \setminus T^{\leqq}_{\tilde G}(i); \; l=1, \ldots, r)$$

is the generating vertex set of a clique of $\tilde{B}_i$ or if at least two vertices of this set are not connected in S_i^*. In both cases either $\{j_{i_l+1}, \ldots, j_r\}$ or $\{j_{i_r+1}\}$ is the next potential set for the clique test according to whether $i_l+1 \neq i_{l+1}$ for some lowest $l<r$ or $i_l+1=i_{l+1}$ $(l=1, \ldots, r-1)$ and $i_r<k-1$.

6.2.5 By Theorem 3.1.1, $Y := \{j_{i_1}, \ldots, j_{i_r}\}$ is the vertex set of a clique of S_i^* iff $Y = \bigcap_{l \in Y} N_{S_i^*}(l)$. Using the representation $M(S_i^*)$ of S_i^*, the test of this condition is equivalent to the Boolean test of $Y^* = \bigwedge_{l \in Y} v_{S_i^*}(l)$, where Y^* is a binary vector of length $|S_i^*|$ containing a 1 at the i_e-th place $(e=1, \ldots, r)$ and 0 otherwise.

6.2.6 Symbolic code of the algorithm K-CAL 1. Let FE, VS defined as in 6.2.3. VS^* denotes the vector consisting of the components of VS without the last element LE, $VS=(FE=x_1, \ldots, x_n=LE)$; $VS^*=(FE, x_2, \ldots, x_{n-1})$.

Algorithm K-CAL 1:



A. *procedure* K-CAL 1:

```
begin
  $l \leftarrow j_l$
  comment $j_l$ is an element of the vertex set of $S_i^*$
  $VS \leftarrow k$
  SUB 2
  SUB 1
  while $EL \neq k$ do SUB 1
end
```

```
B. procedure SUB 1:
   begin
     $EL \leftarrow 1$
     $VS \leftarrow EL, VS$
     if COND 1 and not COND 2 then BLOCK 1
   end
```

```
C. procedure SUB 2:
   begin
     BLOCK 2
     while $EL \neq FE$ do
       begin
         $VS \leftarrow EL, VS$
         if COND 1 then
           begin
             if COND 2 then SUB 1
             else BLOCK 1
           end
         else
           begin
             if $FE = 1$ then SUB 1
           end
         BLOCK 2
       end
   end
```

```
D. procedure COND 1:
   comment Adjacency test
   if $FE$ connected with all $x \in VS^*$ then return true else return false
```

```
E. procedure COND 2:
   comment Clique test
   if $VS^* \neq \bigcap_{l \in VS^*} N_{S_l^*}(l)$ then return true else return false
```

```
F. procedure BLOCK 1:
   begin
     form $K$ from $VS^*$ by substituting $l \rightarrow j_l$
     write $K$
```

```
G. procedure BLOCK 2:
   comment Reduction
   begin
     delete $FE$ from $VS$
     $EL \leftarrow EL + 1$
   end
```

6.3 The Algorithm K-CAL 2

6.3.1 Similar to the method of [2], the algorithm K-CAL 2 is essentially an enumerative tree search algorithm which iteratively works on $M(S_i^*)$.

6.3.2 At some stage n during the algorithm, a complete set of vertices $V_n \subset N_{\tilde{G}}^{\leq}(i) \setminus T_{\tilde{G}}^{\leq}(i)$ is augmented by an other suitably chosen vertex of $N_{\tilde{G}}^{\leq}(i) \setminus T_{\tilde{G}}^{\leq}(i)$ to generate a complete set V_{n+1} at stage $n+1$. If no further augmentation is possible, V_n becomes the vertex set of a clique of S_i^*.

6.3.3 During the tree search, at every stage n one defines three different sets of vertices of S_i^* whose elements, added to V_n, generate a complete set V_{n+1}, namely:

S_n, T_n the set of those vertices of $N_{\tilde{G}}^{\leq}(i) \setminus T_{\tilde{G}}^{\leq}(i)$ which have (not) been used to augment V_n, respectively.

U_n the subset of all elements of R_i which are connected with each element $x \in V_n$.

6.3.4 By a forward branching process choosing $j_{i_n} \in T_n$ from V_n, T_n, S_n, U_n new sets

$$V_{n+1} = V_n \cup \{j_{i_n}\};\ T_{n+1} = T_n \setminus (\Gamma_{T_n}(j_{i_n}) \cup \{j_{i_n}\})$$
$$S_{n+1} = S_n \setminus \Gamma_{S_n}(j_{i_n});\ U_{n+1} = U_n \setminus \Gamma_{U_n}(j_{i_n})$$

are constructed, where

$$\Gamma_Y(j_{i_n}) = \{j \in Y / j \underset{S_i^*}{\not\sim} j_{i_n}\},\quad Y = S_n, T_n, U_n.$$

6.3.5 In a backtracking step during the algorithm (see H. in 6.3.8), j_{i_n} is removed from V_{n+1} reproducing V_n and by the removal of j_{i_n} from the old set T_n and its addition to the old set S_n, new sets T_n and S_n are constructed.

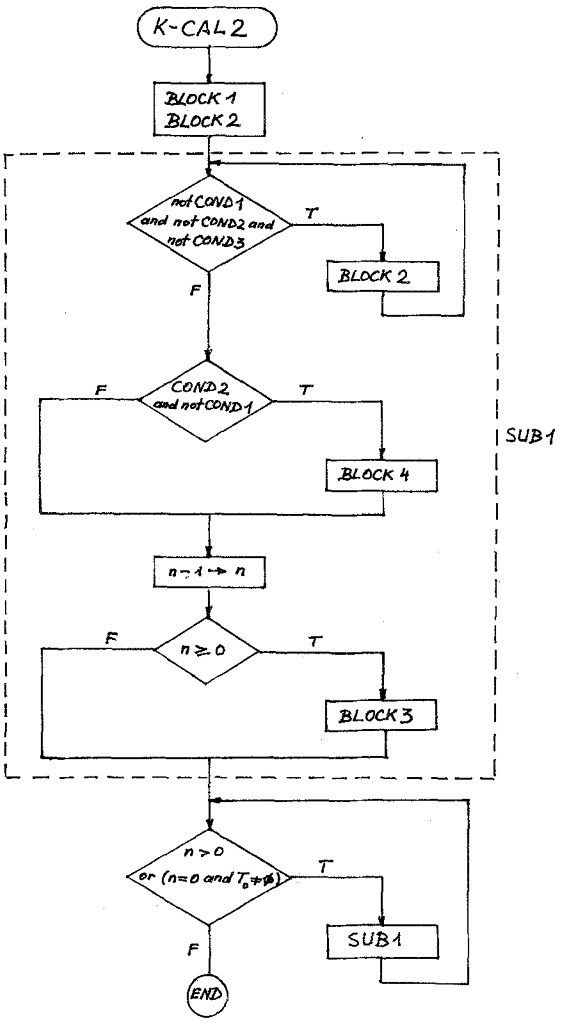
6.3.6 It follows immediately that

$$V_n \in \tilde{B}_i \Leftrightarrow T_n = \emptyset,\ S_n = \emptyset,\ U_n = \emptyset.$$

6.3.7 Further the condition that there exists an element $s \in S_n \cup U_n$ such that $s \underset{S_i^*}{\sim} l$ for all $l \in T_n$ is sufficient for a backtracking process, since, if $s \in S_n$, no maximal complete set containing only elements of $N_{\tilde{G}}^{\leq}(i) \setminus T_{\tilde{G}}^{\leq}(i)$ can result from any forward branching from V_n, and in the case that $s \in U_n$, V_n is a subset of the vertex set of a clique of B_j where $j > i$.

6.3.8 Symbolic code of the algorithm K-CAL 2.

Algorithm K-CAL 2:



```
A. procedure K-CAL 2:
   begin
      BLOCK 1
      BLOCK 2
      SUB 1
   while $n > 0$ or $(n = 0$ and $T_0 \neq \emptyset)$ do SUB 1
   end
```

```
B. procedure SUB 1:
   begin
      while not COND 1 and not COND 2 and not COND 3 do BLOCK 2
      if COND 2 and not COND 1 then BLOCK 4
      $n \leftarrow n-1$
      if $n \geq 0$ then BLOCK 3
   end

C. procedure COND 1:
   comment Branch reduction test
   if $\exists\, x \in S_n \cup U_n : x \underset{S_i^*}{\approx} j$ for all $j \in T_n$ then return true else return false

D. procedure COND 2:
   comment Clique test
   if $S_n = \emptyset \wedge T_n = \emptyset \wedge U_n = \emptyset$ then return true else return false

E. procedure COND 3
   if $T_n = \emptyset \wedge S_n \cup U_n \neq \emptyset$ then return true else return false

F. procedure BLOCK 1:
   comment Initialization of K-CAL 2
   begin
      $n \leftarrow 0$
      $l \rightarrow j_l \in S_i^*$
      $T_0 \leftarrow \{1, \ldots, k-1\}$
      $U_0 \leftarrow \{k, \ldots, |S_i^*|\}$
      $R_i \leftarrow \{k, \ldots, |S_i^*|\}$
      clear $V_0$ and $S_0$
   end

G. procedure BLOCK 2:
   comment Stack generation
   begin
      $S_{n+1} \leftarrow S_n \backslash \Gamma_{S_n}(i_n)$
      $T_{n+1} \leftarrow T_n \backslash (\Gamma_{T_n}(i_n) \cup \{i_n\})$
      $U_{n+1} \leftarrow U_n \backslash \Gamma_{U_n}(i_n)$
      comment $\Gamma_Y(i_n) = \{j \in Y / j \underset{S_i^*}{\nsim} i_n\}$, $Y = S_n, T_n, U_n$
      $V_{n+1} \leftarrow V_n \cup \{i_n\}$
      comment $i_n$ is the first element listed in $T_n$
      $n \leftarrow n+1$
   end

H. procedure BLOCK 3:
   comment Backtrack
   begin
      $V_n \leftarrow V_{n+1} \backslash \{i_n\}$
      $T_n \leftarrow T_n \backslash \{i_n\}$
      $S_n \leftarrow S_n \cup \{i_n\}$
   end
```

```
I. procedure BLOCK 4:
   comment Clique decoding and clique report
   begin
      form $K$ from $V_n$ by substituting $l \leftarrow j_l$
      write $K$
   end
```

D. Theoretical Background of the Algorithm A 2[3]

7. *The Algorithm A 2*

The method described in the following is based on n[4] iterations of a special tree search algorithm, which always operates on a system of sets

$$\mathfrak{M}_i = \{M_1^{(i)}, \dots, M_{s_i}^{(i)}\}, \; i = 1, \dots, n$$

and by which a well defined subset L_i of cliques of $G=(V,E)$ (see 7.1) will be determined. Each of these sets $M_j^{(i)}$ thereby consists of a special subset S_j of $N_G(i) \backslash N_G^{\leqq}(i)$ such that the vertices of S_j are not connected with certain other vertices of G. The strategy, fundamental for the total searching algorithm, depends on the essential statement, that a set C of vertices of G is a clique of L_i iff the relations

$$C \cap M_j^{(i)} = \emptyset \quad \text{for all } \; j = 1, \dots, s_i$$

hold.

7.1 Notations and Definitions

7.1.1 In this part let $G=(V,E)$ be a nondirected (irreflexive) graph in the sense of A. 1.1 represented by its adjacency matrix $M(G)$. Furthermore, we use the notations and definitions of chapter A. and the following two additional definitions:

7.1.2 A clique containing the vertex i $(1 \leq i \leq n)$ but no vertex $j < i$ is called an *i-clique* and the totality L_i of those i-cliques an *i-block*[5].

7.1.3 The *i-significant domain* $M(i)$ is the totality of all those vertices $\neq i$, which are contained in i-cliques.

7.1.4 According to these definitions, we obviously obtain: For each clique $C = \{i, j_1, \dots, j_r\} \in L_i : j_k \in N_G(i) \backslash N_G^{\leqq}(i)$ for all $k = 1, \dots, r$; that means $N_G(i) \backslash N_G^{\leqq}(i)$ is the i-significant domain $M(i)$.

---

[3] This algorithm has been developed by the second author.

[4] n is the number of vertices of the given graph G.

[5] Note that an i-block is different from the "i-system of cliques" in the preceeding chapters, but dual in a certain sense.

7.2 *Theoretical Foundations of Algorithm A 2*

For each i $(1 \leq i \leq n)$, let $M(i) = N_G(i) \setminus N_G^{\leq}(i)$ be the i-significant domain. Let further $\kappa_1, \ldots, \kappa_r; \kappa_{r+1}, \ldots, \kappa_s$ $(\kappa_j \leq \kappa_k$ for $j \leq k)$ be the ordered set of all these vertices of $N_G(i)$. Additionally assume that $\kappa_\lambda \in N_G^{\leq}(i)$ for $\lambda = 1, \ldots, r$ and $\kappa_\mu \in M(i)$ for $\mu = r+1, \ldots, s$. To each of these vertices there is defined a set $M_{\kappa_\varrho}^{(i)}$ with

$$M_{\kappa_\varrho}^{(i)} := \{j \in M(i) / j \underset{G}{\not\sim} \kappa_\varrho\}, \quad \varrho = 1, \ldots, s^{6} \tag{7.1}$$

The system consisting of these s sets will be fundamental for the generation of the i-cliques. For there holds the

Theorem 7.2.1: (*Main theorem for A 2*): *Let $T \subseteq M(i)$ be a set of pairwise connected vertices of G with $i \notin T$. Necessary and sufficient for $C = T \cup \{i\}$ to be an i-clique are the conditions*

$$T \cap M_{\kappa_\varrho}^{(i)} \neq \emptyset \quad \textit{for all} \quad \varrho = 1, \ldots, s.$$

Proof: $\Rightarrow$: We suppose that $C = \{i, c_1, \ldots, c_t\}$ is an i-clique. According to the definition, then the vertices $c_1, \ldots, c_t$ must belong to the i-significant domain $M(i)$ and above all the conditions

$$T \cap M_{\kappa_\varrho}^{(i)} \neq \emptyset \quad \text{for} \quad \varrho = 1, \ldots, r$$

must be fulfilled. For if it would be not so, there must exist an index $\varrho' \leq r$, so that $T \cap M_{\kappa_{\varrho'}}^{(i)} = \emptyset$ holds. But because $M_{\kappa_{\varrho'}}^{(i)}$ contains only those vertices connected with i which are not connected with $\kappa_{\varrho'}$ the assumption $T \cap M_{\kappa_{\varrho'}}^{(i)} = \emptyset$ implies that C cannot contain any vertex which is not connected with $\kappa_{\varrho'}$. That means that $\kappa_{\varrho'}$ would be a vertex which is connected with i and all the vertices $c_1, \ldots, c_t$, i.e. $\kappa_{\varrho'}$ must be an element of C. But also $\kappa_{\varrho'} < i$ therefore C cannot be an i-clique in contradiction to our assumption.

Now we have still to prove, that the conditions of the main theorem are also valid for the sets $M_{\kappa_\varrho}^{(i)}$ with $\varrho = r+1, \ldots, s$. Associated to each element $c_j \in C$ $(1 \leq j \leq t)$ is the set $M_{c_j}^{(i)}$, which is a member of the system (7.1). Because, from footnote 6, c_j is contained in $M_{c_j}^{(i)}$, the conditions of the main theorem are trivially satisfied for all those sets $M_{c_1}^{(i)}, \ldots, M_{c_t}^{(i)}$ of (7.1) which are characterized by the elements $c_1, \ldots, c_t$ of C. Now let $c \notin C$ be an arbitrary element of G connected with i. Then there must exist in C an element c_j, for which $c_j \underset{G}{\not\sim} c$ holds, this means $c_j \in M_c^{(i)}$ and consequently $C \cap M_c^{(i)} \neq \emptyset$. Herewith the first part of the Theorem 7.2.1 has been proved.

$\Leftarrow$: We now assume that the conditions of the theorem are fulfilled for T with $T = \{c_1, \ldots, c_t\}$. Proving the assertion indirectly, we now suppose $C = T \cup \{i\}$ is not yet a clique. Then there must exist a vertex $c \neq i$, $c \notin T$ with $c \underset{G}{\sim} i$ and $c \underset{G}{\sim} c_j$ $(j = 1, \ldots, t)$. According to the definition, $M_c^{(i)}$ contains — besides c — only those elements which are not connected with c, i.e. the vertices $c_1, \ldots, c_t$ cannot belong to $M_c^{(i)}$. Hence, in contradiction to the assumption, $T \cap M_c^{(i)} = \emptyset$ must

[6] Note, that for all ϱ with $r+1 \leq \varrho \leq s$ this set is not empty, because it contains, according to the irreflexivity, the vertex κ_ϱ itself.

hold. Therefore the assumption that $C = T \cup \{i\}$ is not a clique was false. This completes the proof.

Helpful for the construction of an i-clique is the following Lemma, which is an immediate consequence of the main theorem. Using the notation of (7.1), we have

Lemma 7.2.2: *Let T' be a proper subset of an i-clique. Let further Σ ($\Sigma \subseteq S = \{1, \ldots, s\}$) be the full set of those indices σ ($1 \leq \sigma \leq s$) for which*

(*) $T' \cap M_{\kappa_\sigma}^{(i)} = \emptyset$, *if* $\sigma \in \Sigma$, *but*

(**) $T' \cap M_{\kappa_\tau}^{(i)} \neq \emptyset$, *if* $\tau \in S \backslash \Sigma$

holds. Then $C = T \cup T' \cup \{i\}$ is an i-clique iff $T \bigcap M_{\kappa_\sigma}^{(i)} \neq \emptyset$ for all $\sigma \in \Sigma$.

Proof: According to the main theorem, $C = T \cup T' \cup \{i\}$ is an i-clique iff

$$(T \cup T') \cap M_{\kappa_\varrho}^{(i)} \neq \emptyset \quad \text{for all } \varrho = 1, \ldots, s \tag{7.2}$$

holds.

Because of supposition (**), this relation is satisfied for all indices $\varrho \in S/\Sigma$, by regarding (*) therefore, (7.2) can be reduced to $T \cap M_{\kappa_\varrho}^{(i)} \neq \emptyset$ for all $\sigma \in \Sigma$.

This lemma states that for the determination of all i-cliques containing $T' = \{a_1, \ldots, a_r\}$ only the sets $M_{\kappa_\sigma}^{(i)}$ ($\sigma \in \Sigma$) — and only these — are necessary.

Now let a and b be two arbitrary vertices of G, both contained in the same set $M_{\kappa_\varrho}^{(i)}$ of (7.1) with some ϱ ($1 \leq \varrho \leq s$). Further, let us assume that all i-cliques containing $T' \cup \{a\}$ have been determined. If b is not connected with a, then each i-clique containing $T' \cup \{b\}$ cannot contain the vertex a. Quite different however is the situation if $b \underset{G}{\sim} a$. In this case, there may exist i-cliques, which contain, besides $T' \cup \{b\}$, also the vertex a. By the following lemma it is possible to decide if and when this case occurs.

Lemma 7.2.3: *Suppose $T' = \{a_1, \ldots, a_v\}$ fulfills the suppositions of Lemma 7.2.2. Suppose two vertices a, b of G also fulfill the following conditions: $a \underset{G}{\sim} b$, $a, b \underset{G}{\sim} a_j$ ($j = 1, \ldots, v$), $a, b \in M_{\kappa_\sigma}^{(i)}$ ($M_{\kappa_\sigma}^{(i)}$ is one of the sets of (*)). Then an i-clique C, containing $T' \cup \{b\}$, does not contain the element a iff the condition*

$$(M_a^{(i)} \cap C) \backslash \{a\} \neq \emptyset$$

is fulfilled.

Proof: We first have to show that $M_a^{(i)}$ is identical with one of the sets (*) of Lemma 7.2.2. If $M_a^{(i)} \cap T' \neq \emptyset$ holds, then due to $a \notin T'$ (according to assumption), T' must contain at least one element a_λ ($\neq a$) ($1 \leq \lambda \leq v$) with $a_\lambda \in M_a^{(i)}$, i.e. $a_\lambda \underset{G}{\not\sim} a$ in contradiction to the assumption.

Now let C be an i-clique containing $T' \cup \{b\}$ but not containing the vertex a. Then according to the main theorem, the condition of Lemma 7.2.3 must be fulfilled. On the other hand, if the condition of this lemma is fulfilled, C must contain at least one element, which is not connected with a in G.

The algorithm for computing all cliques depends on the successive determination of all i-blocks ($i = 1, \ldots, n$). Thereby, each of these blocks is constructed from the characteristic sets $M_{\kappa_1}^{(i)}, \ldots, M_{\kappa_s}^{(i)}$ of (7.1) by using the main theorem. If one of these sets is empty, then i-cliques do not exist. Otherwise, a searching algorithm on this system of sets will be started. Lemma 7.2.3 guarantees that a clique will never be determined for a second time.

7.3 Remarks

Numerous comparing test series have indicated that the method based on Theorem 7.2.1 of section 7 is well suitable for the treatment of large sparse adjacency matrices, while it is not so efficient for those graphs with a very large number of edges (see the tables of chapter F), therefore we omit the detailed description of the implementation of the algorithm $A2$.

In common with the algorithm $A1$, the method $A2$ has many characteristic properties which are described in the following chapter E.

E. Significance of the Algorithms

8. Characteristic Properties of the Algorithms

8.1 Calculation of Special Clique Sets

In the current section, let $v = (i_1, \ldots, i_s)$ be a vector of labeled vertices of a graph $G = (V, E)$. Then, by the structure of the algorithms developed, it is possible to calculate efficiently the following special clique sets:

(i) the subset $L_{G,v}^{\vee} \subseteq L_G$ of all cliques of G containing *at least* one vertex of v;
(ii) the subset $L_{G,v}^{\wedge} \subseteq L_G$ of all cliques of G which contain all vertices of v, provided that $s \geq 2$ and any two vertices of v are connected with each other.

8.1.1 Computational determination of $L_{G,v}^{\wedge}$. Making use of the algorithm $A1$, the set $L_{G,v}^{\vee}$ can be determined by calculating the i_k-systems B_{i_k} $(k = 1, \ldots, s)$ *relative* to the matrix $\overline{M(G)}$ which can be obtained from the adjacency matrix $M(G)$ of G by exchanging the rows and columns $v_G(i_k)$, $v_G^T(i_k)$ of $M(G)$ with $v_G(n-s+k)$, $v_G^T(n-s+k)$, $(k = 1, \ldots, s)$, respectively. Using algorithm $A2$, it is possible to calculate $L_{G,v}^{\vee}$ by restricting $A2$ to the first s rows and columns of the matrix $\overline{M(G)}$ which can be obtained from $M(G)$ of G by the exchange $v_G(i_k) \leftrightarrow v_G(k)$, $v_G^T(i_k) \leftrightarrow v_G^T(k)$, $(k = 1, \ldots, s)$.

8.1.2 Computational determination of $L_{G,v}^{\vee}$. Let $\overline{M(G)}$ be the matrix obtained from $M(G)$ by exchanging $v_G(i_k)$, $v_G^T(i_k)$ with $v_G(n)$, $v_G^T(n)$, respectively, and where $i_k \in v$ is an element for which $|N_G(i_k)| = \min_{j \in v} |N_G(j)|$. Then using algorithm $A1$, $L_{G,v}^{\wedge}$ is determined by the i_k-system B_{i_k}, which is calculated from the adjacency matrix $M(S'_{i_k})$ (instead of $\overline{M(S_{i_k})}$, see procedure C. of CLIQUE) and where S'_{i_k} is the subgraph of S_{i_k} which is generated by the vertex set

$$N'_G(i_k) = \{j \in N_G(i_k) / j \underset{G}{\sim} i_s \text{ for all } i_s \in v\}.$$

$\overline{M(S'_{i_k})}$ can be obtained from $M(G)$ by deleting those rows $v_G(j)$ and columns $v_G^T(j)$ of $M(G)$ which correspond to the vertices j of G satisfying the condition $j \notin N_G(i_k)$ or $j \in N_G(i_k)$ and $j \underset{G}{\not\sim} i_s$ for some $i_s \in v$.

By an equivalent calculation, $L_{G,v}^{\wedge}$ can be obtained by application of $A\,2$, if $v_G(i_k)$, $v_G^T(i_k)$ is exchanged with $v_G(1)$, $v_G^T(1)$, respectively.

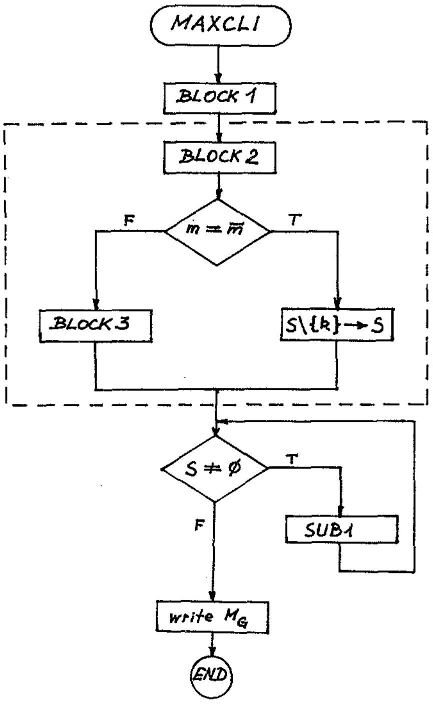
8.1.3 Importance of $L_{G,v}^{\vee}$, $L_{G,v}^{\wedge}$ *for other algorithms.* From a special (heuristic) viewpoint, the calculation of subsets of L_G as $L_{G,v}^{\vee}$, $L_{G,v}^{\wedge}$ is of great importance for the development of efficient cluster algorithms which operate on clique sets. Since the clique set of a graph exponentially grows with the number of vertices of G, the determination of clique subsets of G containing selected vertices of the graph is always desirable. But, moreover, experience shows, that in the case where G is a graph associated with a complex organization structure of more than 200 entities and more than 50% relationships between them, the restriction of clique calculation to special selected vertices of G is necessary for time- and space-saving algorithms.

8.2 Determination of all Cliques of G of Maximal Length

8.2.1 Let $G=(V,E)$ be a reflexive graph, $|V|=n$ and $k=\max_{i\in V}|N_G(i)|$. Further let $A(m)$ $(1\leq m\leq k)$ be the number of those vertices $i\in V$ for which $|N_G(i)|\geq m$. Then it is obvious that there exists a clique $C=(V',E_{V'})$ of G having $|V'|=m$ vertices only if $A(m)\geq m$. In this case $S_{G,m}:=\{j\in V/|N_G(j)|\geq m\}$ is the potential set of all vertices of G for which cliques of length m can be expected. Moreover, in the case of existence, the union $\bigcup_{i\in S_{G,m}} B_i$ of special i-systems contains all cliques of length m.

8.2.2 Using the necessary condition of 8.2.1, the set M_G of all cliques of maximal length can be calculated by an iterative procedure by the following algorithm:

Algorithm MAXCLI:



```
A. procedure MAXCLI:
   begin
     BLOCK 1
     SUB 1
     while $S \neq \emptyset$ do SUB 1
     write $M_G$
   end

B. procedure SUB 1:
   begin
     BLOCK 2
     if $m = \bar{m}$ then $S \leftarrow S \setminus \{k\}$ else BLOCK 3
   end

C. procedure BLOCK 1:
   begin
     $m^* \leftarrow \max \{l/1 \leq l \leq k \wedge A(l) \geq l\}$, $k = \max_{i \in V} |N_G(i)|$
     $m \leftarrow m^*$
     $S \leftarrow S_{G,m} = \{j \in V \mid |N_G(j)| \geq m\}$
     $T \leftarrow \emptyset$
   end
```

```
D. procedure BLOCK 2:
   begin
      Take $k \in S$ and calculate the $k$-system $B_k$,
      storing in $M_G$ only cliques of current maximal length $\bar{m}$;
      $T \leftarrow T \cup \{k\}$;
   end

E. procedure BLOCK 3:
  begin
     $m \leftarrow \bar{m}$
     $S \leftarrow S_{G,m} \backslash T$
  end
```

9. Parallel Computation

9.1.1 All algorithms (known to the authors) for determining the set L_G of all cliques of a finite graph $G=(V, E)$ operate on the complete adjacency matrix $M(G)$ of G. These clique determination algorithms can be divided into two classes. One of them includes the space-saving and more or less time-efficient algorithms, which can forget every clique after its construction. The other class contains those algorithms, which successively build up every clique of G so that long lists of complete but not maximal complete subgraphs of G have to be stored in the computer.

9.1.2 The algorithms $A\,1$, $A\,2$ of this paper belong to the first class. But, moreover, the basic concept of the independent blockwise generation of the i-systems by $A\,1$ (i-blocks by $A\,2$) allows us to calculate these special clique sets by parallel operating processors, so that even in the worst cases a space- and time-saving calculation is possible. Especially, the time efficiency of the algorithms is of great importance if interactive systems are used for cluster algorithms which operate on large clique sets.

F. Test Results and Comments

10. Calculation of L_G for Series of Characteristic Graphs

10.1 The algorithms $A\,1$ (both versions CLIQUE/K-CAL 1 and CLIQUE/K-CAL 2) and $A\,2$ have been written in FORTRAN IV and implemented for the computer SIEMENS 4004/151. $A\,1$ and $A\,2$ were tested and compared with the ALGORITHM 457 of CACM [2] which is the most efficient method among many other clique determination algorithms which have also been tested and compared with $A\,1$ and $A\,2$ [7].

[7] We intend to publish a further paper containing the results of comparison of the tested algorithms.

10.2 Random Graphs

10.2.1 The algorithms $A\,1, A\,2$ were tested for many series of random graphs $R\,(n, k)$, where n is a fixed number of vertices of the graph and k an increasing parameter, which corresponds to the number e of edges of the considered graph. Starting with $k=0$ ($R\,(n, k)$ consists of n isolated vertices) and ending with $k=100$ ($R\,(n, k)$ is the complete graph S_n of n vertices) every series was calculated for increasing k in % relative to S_n ($k\,(S_n)=100\%$). In comparsion to ALGORITHM 457, the methods $A\,1$, version 2 (CLIQUE/K-CAL 2) and $A\,2$ show the same characteristic mode of acting and the (k, t)-diagrams are quite similar. This is obvious, since the tree search procedures used are similar to each other. But because the algorithm $A\,1$, version 1 (CLIQUE/K-CAL 1) needs a binary tree search which is defined by a dual ordering and a different neighborhood clique test, its (k, t)-diagram is quite different from the (k, t)-diagrams of the other algorithms.

But characteristic for all compared methods is that the computing time t exponentially increases with the augmentation of edges and that the maximal computing time is reached between 85% and 90% of edges relative to S_n ($k\,(S_n)=100\%$).

Further, the test series show that in the case of relatively large n ($100 \le n \le 250$) and relatively small k ($k \le 50$) the algorithms $A\,1$ and $A\,2$ are very efficient.

10.2.2 Time-tables and (k, t)-diagrams for $R\,(36, k)$, $0 \le k \le 100$ and $R\,(250, k)$, $k=1, 2, 3, 5, 10$.

e = number of edges of R, N = number of cliques of R, k = density of edges of R in % rel. to S_n, t = computing time in [sec].

$R\,(36, k)$, $0 \le k \le 100$						
e	k	N	ALG. 457 t	$A\,1$/vers. 1 t	$A\,1$/vers. 2 t	$A\,2$ t
0	0	36	0.54	0.48	0.47	0.53
63	10	55	0.84	0.51	0.80	0.78
126	20	80	1.36	1.07	1.44	1.49
189	30	116	1.84	1.59	2.11	2.31
252	40	192	3.16	2.93	3.54	4.66
315	50	294	4.84	5.11	5.20	10.30
378	60	505	8.84	11.44	10.15	25.30
441	70	853	16.30	32.79	19.07	53.28
473	75	1289	26.95	79.13	29.35	83.20
504	80	1827	40.63	143.71	47.81	169.83
536	85	2076	49.32	273.03	58.66	176.31
567	90	3242	81.29	1131.12	96.24	231.02
599	95	4092	109.12	881.21	119.48	194.20
617	98	64	2.27	3.25	2.40	2.73
630	100	1	0.26	0.24	0.14	0.24

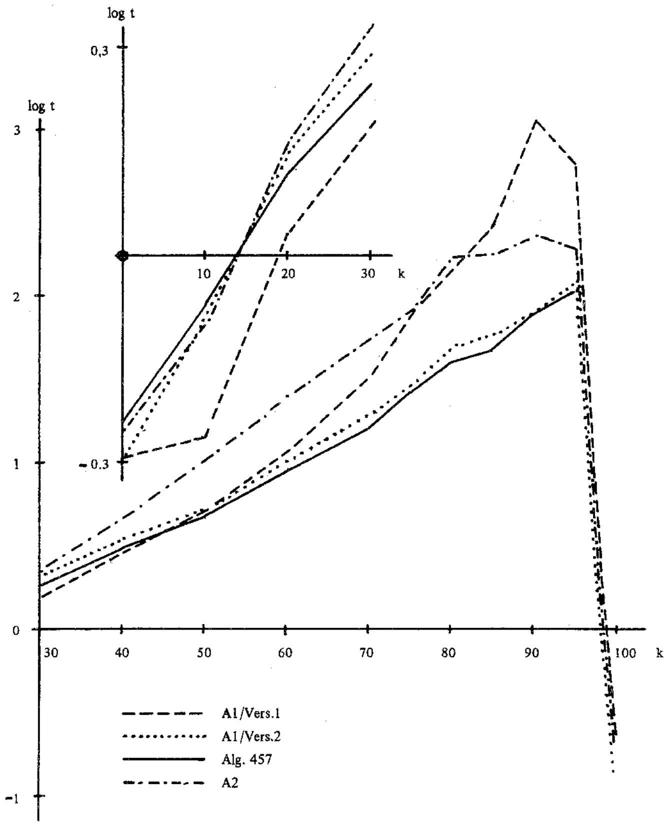


Fig. 1

$R(250, k)$, $k=1, 2, 3, 5, 10$						
e	k	N	ALG. 457 t	A 1/vers. 1 t	A 1/vers. 2 t	A 2 t
311	1	317	17.58	7.40	13.49	14.00
623	2	589	20.94	10.43	18.01	17.45
934	3	815	23.88	13.63	21.06	21.50
1556	5	1157	31.18	19.27	30.03	30.14
3113	10	2456	61.27	47.70	67.95	82.67

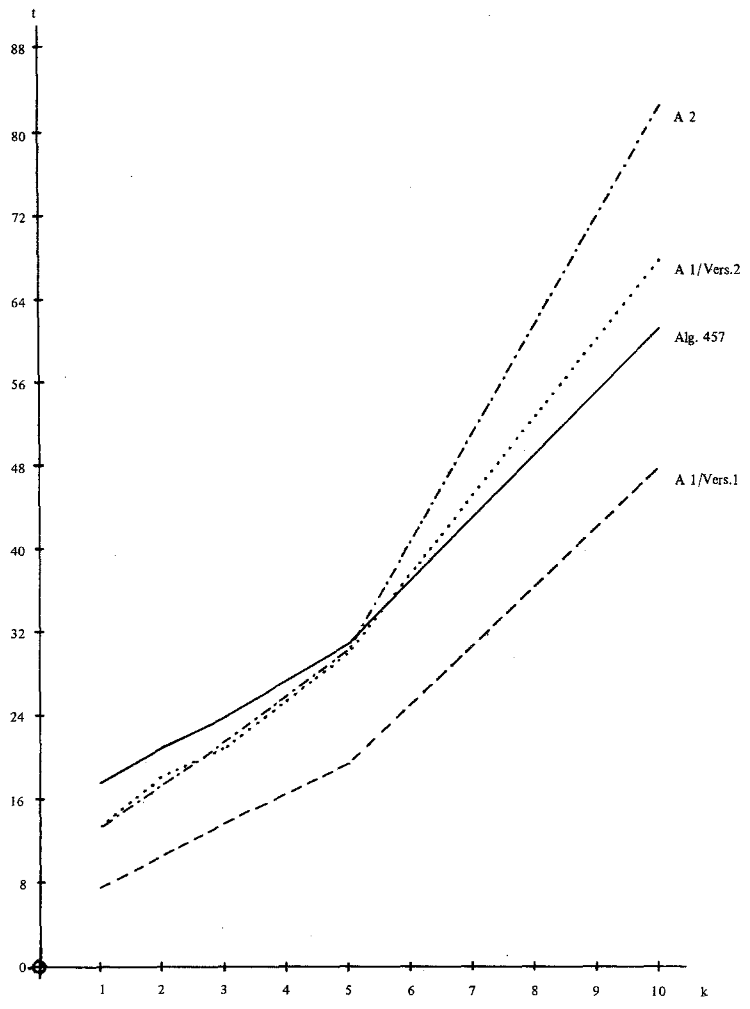


Fig. 2

10.3 Moon-Moser Graphs [10]

10.3.1 There exist three series of graphs $M(n)$ $(n \geq 2)$ with n vertices and with a maximum number $f(n)$ of cliques. It follows from the theory that $f(n) = 3^{n/3}$, $4 \cdot 3^{[n/3]}$, $2 \cdot 3^{[n/3]}$ if $n \equiv 0\,(3)$, $\equiv 1\,(3)$, $\equiv 2\,(3)$, respectively. The graphs $M(n)$ can be obtained, if the set of n vertices is divided into subsets such that as many as possible of them have three vertices and the remaining one have two or four vertices and if two vertices are joined exactly if they do not belong to the same subset.

10.3.2 Time-table of the Moon-Moser series:

$M(n)$, $n \equiv 0\,(3)$, $3 \leq n \leq 27$						
e	k	N	ALG. 457 t	$A\,1$/vers. 1 t	$A\,1$/vers. 2 t	$A\,2$ t
0	0	3	0.11	0.02	0.13	0.04
9	60	9	0.08	0.07	0.08	0.15
27	75	27	0.29	0.24	0.28	0.43
54	82	81	1.14	1.05	1.12	1.40
90	86	243	3.17	3.37	3.02	4.46
135	88	729	10.34	12.02	9.69	14.26
189	90	2 187	34.23	47.42	32.82	46.02
252	91	6 561	105.65	186.42	107.43	151.49
324	92	19 683	354.37	734.96	335.51	447.85

10.3.3 The Moon-Moser test series and many other calculations for graphs with high degree of symmetries confirme the assumption, that the algorithms developed, especially A 1, are very efficient for symmetrically structured graphs.

11. Test Results for the Calculation of $L^{\vee}_{G,v}$ and $L^{\wedge}_{G,v}$

11.1 This section contains the test results for calculating $L^{\vee}_{G,v}$ and $L^{\wedge}_{G,v}$[8] of the graph series $R(36, k)$, $0 \leq k \leq 100$ of 10.2.2 for the vector $v = (12, 24, 36)$, whose components are selected from the vertex sets of the original graphs.

$L^{\vee}_{G,v}$				$L^{\wedge}_{G,v}$			
k	N	$A\,1$/vers. 2 t	$A\,2$ t	k	N	$A\,1$/vers. 2 t	$A\,2$ t
10	13	0.20	0.19	10	0	0.03	0.03
20	25	0.35	0.53	20	0	0.03	0.03
30	39	0.61	1.12	30	0	0.03	0.03
40	73	1.21	1.96	40	0	0.03	0.03
50	142	2.49	4.85	50	0	0.03	0.03
60	235	4.74	10.39	60	14	0.27	0.37
70	504	13.92	29.69	70	25	0.63	0.84
75	791	20.87	48.45	75	57	1.27	1.68
80	1092	34.07	71.58	80	35	0.98	1.24
85	1552	55.17	126.14	85	99	2.84	5.45
90	2684	114.99	216.30	90	246	8.37	12.55
95	4092	130.94	281.32	95	350	10.89	16.45
98	64	2.19	2.78	98	32	1.09	1.29

[8] It is possible to calculate $L^{\wedge}_{G,v}$ by modification of Algorithm 457.

12. Tables Showing the Blockwise Generation of the Clique Set of a Graph

12.1 The following tables give the computing time for each i-system (i-block) of the random graph R (36, 80) of the series of 10.2.2 calculated by A 1/vers. 2 (A 2), respectively.

The computing time of every i-system (i-block) is related to the number of cliques constructed and from the totality of these computing times it is to be expected that parallel calculation will increase the efficiency of the developed algorithms.

12.2 Blockwise Generation of R (36, 80)

A 1/version 2				*A* 2			
No. *i*-system	*i*	*N*	*t*	No. *i*-block	*i*	*N*	*t*
1	15	6	0.13	1	13	253	18.60
2	24	3	0.05	2	9	272	24.54
3	8	6	0.12	3	16	264	24.68
4	19	12	0.29	4	21	157	14.35
5	3	39	0.81	5	29	101	7.78
6	33	26	0.63	6	14	157	19.99
7	20	7	0.22	7	17	99	6.97
8	34	18	0.47	8	30	67	4.96
9	31	24	0.54	9	2	78	13.34
10	25	42	0.97	10	11	127	15.80
11	11	95	2.45	11	7	69	3.93
12	7	101	2.17	12	20	16	1.61
13	2	78	2.38	13	25	38	1.66
14	17	99	2.38	14	31	19	0.67
15	30	67	1.73	15	34	18	1.07
16	14	157	4.22	16	3	53	3.24
17	29	101	2.56	17	19	12	0.87
18	21	157	3.93	18	24	5	0.20
19	16	264	6.83	19	8	7	0.40
20	9	272	6.76	20	33	9	0.27
21	13	253	5.90	21	23	2	0.05
				22	27	2	0.05
				23	28	2	0.03

References

[1] Bednarek, A. R., Taulbee, O. E.: On maximal chains. Roum. Math. Pures et Appl. *11*, 23—25 (1966).

[2] Bron, C., Kerbosch, J.: Finding all cliques of an undirected graph. Collected Algorithms from CACM (Algorithm 457). 1971.

[3] Christofides, N.: An algorithm for the chromatic number of a graph. The Comp. J. *14*, 38 (1971).

[4] Hakimi, S. L., Frank, H.: Maximum internally stable sets of a graph. J. of Math. Anal. and Appl. *25*, 296 (1969).

[5] Harary, F., Ross, I. C.: A procedure for clique detection using the group matrix. Sociometry *20*, 205—215 (1957).

[6] Johnson, L. F.: Determining cliques of a graph. Proc. of the fifth Conf. on Num. Math. *1957*, 429—437.

[7] Leifman, L. J.: On construction of all maximal complete subgraphs (cliques) of a graph. Preprint, Dept. of Math., Univ. Haifa, Israel, 1976.

[8] Meeusen, W., Cuyvers, L.: Clique detection in directed graphs: a new algorithm. J. of Comp. and Appl. Math. *1*, 185—193 (1975).

[9] Maghout, K. H.: Sur la détermination des nombres de stabilité et du nombre chromatique d'un graph. C. r. Acad. Sci. (Paris) *248*, 3522—3523 (1959).

[10] Moon, J. W., Moser, L.: On cliques in graphs. Israel J. Math. *3*, 23—28 (1965).

[11] Peay, E. R., jr.: An iterative clique detection procedure. Michigan Math. Psychol. Program: MMPP 70-4 (1970).

[12] Tinhofer, G.: Methoden der angewandten Graphentheorie (Kapitel 2, Algorithmus 8). Wien-New York: Springer 1976.

Dr. L. Gerhards, Dr. W. Lindenberg
Institut für Mathematik der
Gesellschaft für Mathematik
und Datenverarbeitung mbH
Schloß Birlinghoven
D-5205 St. Augustin 1
Federal Republic of Germany